

In the United States Patent and Trademark Office  
Board of Patent Appeals and Interferences

Appeal Brief

In re the Application of:

Kevin Brown and Raghupathi Keshava Murthy

Serial No. 10/675,265

Filed: September 29, 2003

Attorney Docket No. SVL920030045US1

METHOD, SYSTEM, AND PROGRAM FOR  
PREDICATE PROCESSING BY ITERATOR FUNCTIONS

Submitted by:

Janaki K. Davda  
Konrad, Raynes & Victor LLP  
315 So. Beverly Dr., Ste. 210  
Beverly Hills CA 90212  
(310) 556-7983  
(310) 556-7984 (fax)

## TABLE OF CONTENTS

I. Real Party in Interest.....	1
II. Related Appeals, Interferences, and Judicial Proceedings.....	1
III. Status of the Claims .....	1
IV. Status of Amendments .....	1
V. Summary of the Claimed Subject Matter.....	2
VI. Grounds of Rejection to Be Reviewed on Appeal .....	7
VII. Argument .....	7
A. Rejection under 35 U.S.C. 102(e) over Andrei (U.S. Patent No. 6,801,905) .....	7
1. Claims 1-5, 14-18, and 27-31 .....	7
2. Claims 6-11, 19-24, and 32-37 .....	12
3. Claims 12-13, 25-26, and 38-39 .....	15
B. Conclusion .....	17
VIII. Claims Appendix .....	18
IX. Evidence Appendix .....	27
X. Related Proceedings Appendix .....	28

I. Real Party in Interest

The entire right, title and interest in this patent application is assigned to real party in interest International Business Machines Corporation.

II. Related Appeals, Interferences, and Judicial Proceedings

Appellant, Appellant's legal representative, and Assignee are not aware of any other prior or pending appeals, interferences, and judicial proceedings which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. Status of the Claims

Claims 1-39 are pending.

Claim 1 has been objected to.

Claims 1-39 have been rejected in view of prior art under 35 U.S.C. 102(e).

The rejection of the claims under 35 U.S.C. 102(e) is being appealed for all pending claims 1-39.

IV. Status of Amendments

In response to a Final Office Action, mailed on February 8, 2007, Applicants filed a "Reply Under 37 CFR 1.116 – Expedited Procedure Request for Reconsideration", submitted on April 9, 2007, in which claim 1 was amended to overcome a claim objection by removing an extraneous space (i.e., "result s" was changed to "results").

An Advisory Action mailed on April 17, 2007 did not indicate whether or not the minor amendment to claim 1 would be entered. For purposes of appeal, Applicants are assuming that the minor amendment has not been entered. Applicants are not appealing the objection, but Applicants would like to request that the minor amendment to claim 1 be entered by Examiner Timblin at the time he submits an Examiner's Answer.

A Notice of Appeal and Pre-Appeal Brief Request for Review was filed on May 8, 2007.

A Notice of Panel Decision from Pre-Appeal Brief Review maintains the rejection of claims 1-39.

V. Summary of the Claimed Subject Matter

Claim 1 describes a method for processing predicates in an iterator function (e.g., Specification, page 4, paragraph 14). Under control of an iterator function processor that executes as part of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), when an iterator function included in a statement is invoked (e.g., Specification, page 10, paragraph 32; page 11, paragraph 35), one or more predicates included in the statement are obtained (e.g., Specification, page 11, paragraph 35; page 12, paragraph 35); the one or more predicates are applied to a row of data (e.g., Specification, page 12, paragraph 35); if applying the one or more predicates results in a match, the row of data is returned (e.g., Specification, page 14, paragraph 43); and, if applying the one or more predicates does not result in a match, another row of data for which application of the one or more predicates result s in a match is searched for (e.g., Specification, page 14, paragraph 44).

Claim 14 describes an article of manufacture including a program for processing predicates in an iterator function, wherein the program causes operations to be performed (e.g., Specification, page 4, paragraph 14; page 6, paragraph 18; Figure 1; page 22, paragraphs 59-60; Figure 5). Under control of an iterator function processor that executes as part of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), when an iterator function included in a statement is invoked (e.g., Specification, page 10, paragraph 32; page 11, paragraph 35), one or more predicates included in the statement are obtained (e.g., Specification, page 11, paragraph 35; page 12, paragraph 35); the one or more predicates are applied to a row of data (e.g., Specification, page 12, paragraph 35); if applying the one or more predicates results in a match, the row of data is returned (e.g., Specification, page 14, paragraph 43); and, if applying the one or more predicates does not result in a match, another row of data for which application of the one or more predicates result s in a match is searched for (e.g., Specification, page 14, paragraph 44).

Claim 27 describes a computer system having at least one program for processing predicates in an iterator function (e.g., Specification, page 4, paragraph 14; page 6, paragraph 18; Figure 1; page 22, paragraphs 59-60; Figure 5). Under control of an

iterator function processor that executes as part of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), when an iterator function included in a statement is invoked (e.g., Specification, page 10, paragraph 32; page 11, paragraph 35), one or more predicates included in the statement are obtained (e.g., Specification, page 11, paragraph 35; page 12, paragraph 35); the one or more predicates are applied to a row of data (e.g., Specification, page 12, paragraph 35); if applying the one or more predicates results in a match, the row of data is returned (e.g., Specification, page 14, paragraph 43); and, if applying the one or more predicates does not result in a match, another row of data for which application of the one or more predicates results in a match is searched for (e.g., Specification, page 14, paragraph 44).

Claims 2, 15, and 28 describe, wherein obtaining the one or more predicates comprises obtaining a qualification descriptor that describes the one or more predicates and one or more functions that are used to process the one or more predicates (e.g., Specification, page 11, paragraph 35).

Claims 3, 16, and 29 describe that each function is used to process one of the predicates (e.g., Specification, page 7, paragraph 22; page 12, paragraph 36).

Claims 4, 17, and 30 describe that each of the one or more predicates comprises a simple predicate (e.g., Specification, page 9, paragraph 27; page 11, paragraph 34).

Claims 5, 18, and 31 describe, wherein the iterator function is invoked by the data store engine and further comprising returning the row of data to the data store engine (e.g., page 14, paragraph 43).

Claim 6 describes a method for processing predicates (e.g., Specification, page 4, paragraph 15). Under control of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), a statement including an iterator function and one or more predicates is received (e.g., Specification, page 7, paragraph 21); a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates is created (e.g., Specification, page 7, paragraph 22); and the iterator function is invoked one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied

or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine (e.g., Specification, page 10, paragraph 32 – page 11, paragraph 34).

Claim 19 describes an article of manufacture including a program for processing predicates, wherein the program causes operations to be performed (e.g., Specification, page 4, paragraph 15; page 6, paragraph 18; Figure 1; page 22, paragraphs 59-60; Figure 5). Under control of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), a statement including an iterator function and one or more predicates is received (e.g., Specification, page 7, paragraph 21); a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates is created (e.g., Specification, page 7, paragraph 22); and the iterator function is invoked one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine (e.g., Specification, page 10, paragraph 32 – page 11, paragraph 34).

Claim 32 describes a computer system having at least one program for processing predicates (e.g., Specification, page 4, paragraph 15; page 6, paragraph 18; Figure 1; page 22, paragraphs 59-60; Figure 5). Under control of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), a statement including an iterator function and one or more predicates is received (e.g., Specification, page 7, paragraph 21); a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates (e.g., Specification, page 7, paragraph 22) is created; and the iterator function is invoked one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine (e.g., Specification, page 10, paragraph 32 – page 11, paragraph 34).

Claims 7, 20, and 33 describe that the qualification descriptor provides a handle to each of the one or more functions (e.g., Specification, page 7, paragraph 22).

Claims 8, 21, and 34 describe, when the iterator function is invoked, receiving an indication from the iterator function indicating which of the one or more predicates were applied by the iterator function (e.g., Specification, page 11, paragraph 34; page 13, paragraphs 40, 42).

Claims 9, 22, and 35 describe, when the iterator function is invoked, receiving a row of data from the iterator function that matches the qualification of the one or more predicates (e.g., Specification, page 10, paragraphs 32-33).

Claims 10, 23, and 36 describe applying one or more additional predicates to the received row of data, wherein the one or more additional predicates refer to a column of data that is not in a result set generated by the iterator function (e.g., Specification, page 11, paragraph 34).

Claims 11, 24, and 37 describe applying one or more additional predicates to the received row of data, wherein the one or more additional predicates performs a join between two tables (e.g., Specification page 11, paragraph 34).

Claim 12 describes a method for processing predicates (e.g., Specification, page 4, paragraphs 14 and 15). Under control of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), a statement including an iterator function and one or more predicates is received (e.g., Specification, page 7, paragraph 21); a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates is created (e.g., Specification, page 7, paragraph 22); and the iterator function is invoked (e.g., Specification, page 10, paragraph 32). Under control of an iterator function processor (e.g., Specification, page 6, paragraph 18; Figure 1), the qualification descriptor is retrieved (e.g., Specification, page 11, paragraph 35); a row of data is obtained that matches at least one of the one or more predicates in the qualification descriptor (e.g., Specification, page 14, paragraph 43); indicators are set to indicate which of the one or more predicates have been applied by the iterator function processor (e.g., Specification, page 13, paragraphs 40-42); and the row of data is returned to the data store engine (e.g., Specification, page 14, paragraph

43). Under control of the data store engine, any of the one or more of the predicates included in the statement that have not been applied by the iterator function processor are applied (e.g., Specification, page 11, paragraph 34).

Claim 25 describes an article of manufacture including a program for processing predicates, wherein the program causes operations to be performed (e.g., Specification, page 4, paragraphs 14-15; page 6, paragraph 18; Figure 1; page 22, paragraphs 59-60; Figure 5). Under control of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), a statement including an iterator function and one or more predicates is received (e.g., Specification, page 7, paragraph 21); a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates is created (e.g., Specification, page 7, paragraph 22); and the iterator function is invoked (e.g., Specification, page 10, paragraph 32). Under control of an iterator function processor (e.g., Specification, page 6, paragraph 18; Figure 1), the qualification descriptor is retrieved (e.g., Specification, page 11, paragraph 35); a row of data is obtained that matches at least one of the one or more predicates in the qualification descriptor (e.g., Specification, page 14, paragraph 43); indicators are set to indicate which of the one or more predicates have been applied by the iterator function processor (e.g., Specification, page 13, paragraphs 40-42); and the row of data is returned to the data store engine (e.g., Specification, page 14, paragraph 43). Under control of the data store engine, any of the one or more of the predicates included in the statement that have not been applied by the iterator function processor are applied (e.g., Specification, page 11, paragraph 34).

Claim 38 describes a computer system for processing predicates (e.g., Specification, page 4, paragraphs 14-15; page 6, paragraph 18; Figure 1; page 22, paragraphs 59-60; Figure 5). Under control of a data store engine (e.g., Specification, page 6, paragraph 18; Figure 1), a statement including an iterator function and one or more predicates is received (e.g., Specification, page 7, paragraph 21); a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates is created (e.g., Specification, page 7, paragraph 22); and the iterator function is invoked (e.g., Specification, page 10, paragraph 32). Under control of an iterator function processor (e.g., Specification, page



6, paragraph 18; Figure 1), the qualification descriptor is retrieved (e.g., Specification, page 11, paragraph 35); a row of data is obtained that matches at least one of the one or more predicates in the qualification descriptor (e.g., Specification, page 14, paragraph 43); indicators are set to indicate which of the one or more predicates have been applied by the iterator function processor (e.g., Specification, page 13, paragraphs 40-42); and the row of data is returned to the data store engine (e.g., Specification, page 14, paragraph 43). Under control of the data store engine, any of the one or more of the predicates included in the statement that have not been applied by the iterator function processor are applied (e.g., Specification, page 11, paragraph 34).

Claims 13, 26, and 39 describe that the qualification descriptor describes one or more simple predicates to be applied by the iterator function processor (e.g., Specification, page 9, paragraph 27; page 11, paragraph 34).

#### VI. Grounds of Rejection to Be Reviewed on Appeal

A concise statement listing each ground of rejection presented for review is as follows:

A. Whether Claims 1-39 are anticipated by Andrei (U.S. Patent No. 6,801,905) under 35 U.S.C. 102(e).

#### VII. Argument

A. Rejection under 35 U.S.C. 102(e) over Andrei (U.S. Patent No. 6,801,905)

##### 1. Claims 1-5, 14-18, and 27-31

Anticipation requires that the *identical invention* must be shown in a single reference in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). [Emphasis added.]

Claims 1, 14, and 27 describe, under control of an iterator function processor that executes as part of a data store engine, when an iterator function included in a statement is invoked, obtaining one or more predicates included in the statement, applying the one or more predicates to a row of data, if applying the one or more predicates results in a

match, returning the row of data, and, if applying the one or more predicates does not result in a match, searching for another row of data for which application of the one or more predicates results in a match.

On the other hand, as described in Applicants' Background of the Invention, a conventional iterator function receives a set of arguments and returns a table to the SQL statement that invokes the function (e.g., Specification, page 1, paragraph 4). In particular, the iterator function creates a virtual table with a result set, and then the qualification (i.e., predicate) is applied to the virtual table to filter rows of data in the virtual table. In many cases, only a small percentage of rows of data in the virtual table remain after the qualification is applied, but, unfortunately, because the qualification is applied after the rows of data are retrieved for the result set, many rows of data are unnecessarily retrieved for the virtual table. For example, an application program submitting SQL statement (2) is interested in articles with a score (assigned by the text\_search() iterator function) greater than or equal to 0.9, but, because the text\_search() iterator function is not aware of the score qualification, the text\_search() iterator function returns a virtual table with all articles that have "Bush" in the same sentence as "recession," without applying the predicate for the score.

To avoid this problem, under control of an iterator function processor, when an iterator function included in a statement is invoked, one or more predicates included in the statement are obtained and the one or more predicates are applied to a row of data before the row of data is returned.

In claims 1, 14, and 27, with reference to "when an iterator function included in a statement is invoked", the Examiner cites figure 8A, Col. 11, lines 4-13, and Col. 24, lines 25-67. Figure 8A describes receiving a query that is parsed and normalized and describes adding subplans to an equivalence class. Applicants respectfully submit that Figure 8A does not anticipate "when an iterator function included in a statement is invoked". Col. 11, lines 4-13, describe that SQL statements are passed to the parser which converts the statements into a query tree. Applicants respectfully submit that this does not anticipate "when an iterator function included in a statement is invoked".

At Col. 6, lines 23-27, the Andrei patent describes that a query execution plan or QEP is a demand-driven tuple stream "iterator" tree, which is a data structure that is

interpreted by the relational database management system's execution module or engine. Col. 24, lines 25-67, describe that the physical operators module implements optimization time physical operators that are used for the optimization of the Volcano runtime iterators and describes that they are not themselves runtime iterators that implement the open/next/close protocol, that, rather, the physical operators module contains optimization time descriptions of those runtime iterators which are used for evaluation and costing during the optimization process. Applicants respectfully submit that mention of runtime iterators does not anticipate "when an iterator function included in a statement is invoked".

In the Advisory Action mailed on April 17, 2007, the Examiner states that: "Andrei teaches an operators module that contains optimization time descriptions of those runtime iterates which are used for evaluation during the optimization process (co. 24, lines 29-34). The Examiner respectfully submits that mentioning 'runtime iterators' at least suggests invoking an iterator function (i.e., iterators executed or invoked at runtime)". Applicants respectfully traverse. Anticipation requires that the *identical invention* must be shown in a single reference in as complete detail as is contained in the claims and a mere suggestion is not enough for anticipation ("Mere suggestion from the prior art . . . is not sufficient to constitute anticipation. The prior art in order to be anticipatory must show how to do the thing disclosed." Wm. B. Schaife & Sons Co. v. Falls City Woolen Mills, 209 F. 210, 218 (C.C.A. 6); Canda v. Michigan Malleable Iron Co., 124 F. 486, 492 (C.C.A. 6); Westinghouse Air-Brake Co. v. Great Northern Ry. Co., 88 F. 258, 263 (C.C.A. 2)).

In the Advisory Action, the Examiner also submits that: "Andrei further describes iterators in a query execution plan which is a tree of relational algorithms applied to input tuple streams (col. 6, line 23-40). As the runtime iterator is used for evaluation the claimed limitation of invoking an iterator function is taught." Applicants respectfully traverse. A tree of relational algorithms does not show the identical invention of an *iterator function included in a statement being invoked*.

In the Advisory Action, the Examiner further submits that: "Andrei also teaches invoking access methods (col. 11, line 28-32 and col. 15, line 30-37). Andrei teaches invoking several times an enforcer rule (col. 15, line 34-35) while operators that are used

for optimizing runtime iterators contain, in part, a description of grouping enforcement (col. 24, lines 25-30). Therefore, Andrei teaches when an iterator function is invoked". Applicants respectfully traverse.

The Andrei patent describes that the execution unit generates calls into lower-level routines, such as the access methods, for retrieving relevant information (e.g., row) from the database table (e.g., Specification, Col. 11, lines 28-32). However, such an access method does not show the identical invention of an *iterator function included in a statement being invoked*.

The Andrei patent also describes that invocation of an enforcer rule produces a sorting operation over the cheapest unordered sub-plan (Col. 15, lines 30-37). Such an enforcer rule producing a sort operation does not show the identical invention of an *iterator function included in a statement being invoked and that, when invoked*, when an iterator function included in a statement is invoked, obtains one or more predicates included in the statement; applies the one or more predicates to a row of data; if applying the one or more predicates results in a match, returning the row of data; and, if applying the one or more predicates does not result in a match, searching for another row of data for which application of the one or more predicates result s in a match.

Moreover, the Andrei patent describes the physical operators module that contains optimization time descriptions of those runtime iterators which are used for evaluation and costing during the optimization method (Col. 24, lines 25-35). Such runtime iterators which are used for evaluation and costing during the optimization method does not show the identical invention of an *iterator function included in a statement being invoked and that, when invoked*, when an iterator function included in a statement is invoked, obtains one or more predicates included in the statement; applies the one or more predicates to a row of data; if applying the one or more predicates results in a match, returning the row of data; and, if applying the one or more predicates does not result in a match, searching for another row of data for which application of the one or more predicates result s in a match.

As to obtaining one or more predicates included in the statement, applying the one or more predicates to a row of data, if applying the one or more predicates results in a match, returning the row of data, and, if applying the one or more predicates does not

result in a match, searching for another row of data for which application of the one or more predicates results in a match, Applicants respectfully submit that the Andrei patent does not describe that any of this processing occurs "when an iterator function included in a statement is invoked".

Also, the Andrei patent describes at Col. 11, lines 33-40, that, operating under the control of these instructions, the execution unit generates calls into lower-level routines, such as the access methods, for retrieving relevant information (e.g., row) from the database table, and, after the plan has been executed by the execution unit, the server returns a query result or answer table back to the client(s). Applicants respectfully submit that merely retrieving relevant information does not anticipate "when an iterator function included in a statement is invoked . . . if applying the one or more predicates results in a match, returning the row of data, and, if applying the one or more predicates does not result in a match, searching for another row of data for which application of the one or more predicates results in a match."

Thus, claims 1, 14, and 27 are not anticipated by the Andrei patent.

Dependent claims 2-5, 15-18, and 28-31 incorporate the language of independent claims 1, 14, and 27 and add additional novel elements. Therefore, dependent claims 2-5, 15-18, and 28-31 are not anticipated by the Andrei patent for at least the same reasons as were discussed with respect to claims 1, 14, and 27.

Also, claim 2 describes that, under control of an iterator function processor, when an iterator function included in a statement is invoked (from independent claim 1), obtaining the one or more predicates comprises obtaining a qualification descriptor that describes the one or more predicates and one or more functions that are used to process the one or more predicates. In the Advisory Action, the Examiner submits that: "in figure 5, Andrei shows logical properties computed by the optimizer for a plan (e.g., figure 4). As can be seen, the predicates applied and predicates to apply are shown with their respective functions (i.e., aggregation functs). From the table in figure 5, it can be construed that the aggregation functs are used to process the one or more predicates . . . ". Applicants respectfully traverse. Figure 5 of the Andrei patent describes a table illustrating equivalence class level logical properties computed by the optimizer for a sample plan cache (Col. 17, lines 52-55). Such a table does not show the identical

invention of, when an iterator function included in a statement is invoked, obtaining one or more predicates included in the statement (from claim 1) by obtaining a qualification descriptor that describes the one or more predicates and one or more functions that are used to process the one or more predicates (included in the statement, from claim 1).

Claim 3 depends from claim 2 and describes that each function is used to process one of the predicates. The Examiner cites Col. 31, lines 1-11, as teaching this. The cited portion of the Andrei patent describes a sub-plan, but there is no description of each function being used to process one of the predicates.

Claim 5 describes that the iterator function is invoked by the data store engine and returns the row of data to the data store engine. The Examiner cites Col. 11, lines 33-40 as teaching this. The cited portion of the Andrei patent describes that the execution unit generates calls into lower-level routines, such as the access methods, for retrieving relevant information (e.g., row) from the database table and that after the plan has been executed by the execution unit, the server returns the query result or answer table back to the client. Neither the execution unit retrieving relevant information nor returning the query result to the client shows the identical invention of the iterator function returning a row of data to the data store engine.

Accordingly, it is respectfully submitted that the rejection of claims 1-5, 14-18, and 27-31 as anticipated by the Andrei patent should be reversed.

## 2. Claims 6-11, 19-24, and 32-37

Claims 6, 19, and 32 describe, under control of a data store engine, receiving a statement including an iterator function and one or more predicates, creating a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates, and invoking the iterator function one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine.

The Examiner cites figure 8A of the Andrei patent as teaching receiving a statement including an iterator function and one or more predicates. Figure 8A of the Andrei patent merely describes that a query is received.

The Examiner cites figure 5 as teaching creating a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates. Applicants respectfully submit that figure 5 does not show the identical invention of one or more functions that are to be used to evaluate the one or more predicates included in the received statement (from previous elements of claims 6, 19, and 32).

The Examiner cites figures 5 and 8B and Col. 11, lines 33-40, as teaching invoking the iterator function one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine. Applicants respectfully submit that figure 5 merely mentions "applied predicates" and "predicates to apply", while figure 8B describes determining a final plan for execution of the query and generating a physical operator tree. Col. 11, lines 33-40 describe that, operating under the control of these instructions, the execution unit generates calls into lower-level routines, such as the access methods, for retrieving relevant information (e.g., row) from the database table, and, after the plan has been executed by the execution unit, the server returns a query result or answer table back to the client(s). Applicants respectfully submit that the identical invention of invoking the iterator function one or more times until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine is not shown by the Andrei patent.

Also, because the execution unit generates calls for retrieving information, there is no need for the data store engine to apply predicates included in the statement that have not been applied by the iterator function.

*Applicants respectfully submit that the Andrei patent does not describe that at least one predicate has been applied by the iterator function and that one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine.*

Thus, claims 6, 19, and 32 are not anticipated by the Andrei patent.

Dependent claims 7-11, 20-24, 33-37 incorporate the language of independent claims 6, 19, and 32 and add additional novel elements. Therefore, dependent claims 7-11, 20-24, 33-37 are not anticipated by the Andrei patent for at least the same reasons as were discussed with respect to claims 6, 19, and 32.

Also, claim 10 describes applying one or more additional predicates to the received row of data, wherein the one or more additional predicates refer to a column of data that is not in a result set generated by the iterator function. On the other hand, the cited portion of the Andrei patent in figure 5 and col. 6, lines 23-41 describe "applied predicates" and "predicates to apply" and a query execution plan. There is no description in the Andrei patent of applying one or more additional predicates to the received row of data, wherein the one or more additional predicates refer to a column of data that is not in a result set generated by the iterator function. Instead, the Andrei patent describes that the execution unit generates calls into lower-level routines, such as the access methods, for retrieving relevant information (e.g., row) from the database table (Col. 11, lines 33-40). The Andrei patent does not distinguish between predicates applied by an iterator function and those applied by the data store engine.

Claim 11 describes applying one or more additional predicates to the received row of data, wherein the one or more additional predicates performs a join between two tables. The cited portion of the Andrei patent at Col. 5, lines 9-18 and figure 4, describes that an equivalence class is a grouping of plans or sub-plans and that an example sub-plan joins tables A and B. Merely joining tables does not show the identical invention of applying one or more additional predicates to the received row of data, wherein the one or more additional predicates performs a join between two tables.

Accordingly, it is respectfully submitted that the rejection of claims 6-11, 19-24, and 32-37 as anticipated by the Andrei patent should be reversed.



### 3. Claims 12-13, 25-26, and 38-39

Claims 12, 25, and 38 describe the interaction between a data store engine and an iterator function processor in processing an iteration function. Under control of a data store engine, a statement including an iterator function and one or more predicates is received, a qualification descriptor is created that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates, and the iterator function is invoked. Under control of an iterator function processor, the qualification descriptor is retrieved, a row of data that matches at least one of the one or more predicates in the qualification descriptor is obtained; indicators are set to indicate which of the one or more predicates have been applied by the iterator function processor; and the row of data is returned to the data store engine. Under control of the data store engine, any of the one or more of the predicates included in the statement that have not been applied by the iterator function processor are applied.

Applicants respectfully submit that the elements of claims 12, 25, and 38 are not anticipated by the Andrei patent for at least the reasons discussed with reference to claims 1, 14, 27, 6, 19, and 32.

Moreover, Applicants respectfully submit that the Andrei patent does not describe that an iterator function obtains a row of data that matches at least one of the one or more predicates included in the received statement in the qualification descriptor and sets indicators to indicate which of the one or more predicates have been applied by the iterator function processor and that the data store engine applies any of the one or more of the predicates included in the statement that have not been applied by the iterator function processor. Because the execution unit of the Andrei patent generates calls into lower-level routines, such as the access methods, for retrieving relevant information (e.g., row) from the database table, there is no need for the data store engine to apply predicates included in the statement that have not been applied by the iterator function.

Figure 8A of the Andrei patent describes a query being processed. Receiving a query does not show the identical invention of receiving a statement *including an iterator function and one or more predicates*.

As to, under control of a data store engine, creating a qualification that describes the one or more predicates in the received statement and one or more functions that are to be used to evaluate the one or more predicates and, under control of an iterator function processor, the qualification descriptor is retrieved, figure 5 does not show the identical invention of a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates *included in the received statement*.

Applicants respectfully submit that the Andrei patent does not anticipate the subject matter of claims 12, 25, and 38.

Dependent claims 13, 26, and 39 incorporate the language of independent claims 12, 25, and 38 and add additional novel elements. Therefore, dependent claims 13, 26, and 39 are not anticipated by the Andrei patent for at least the same reasons as were discussed with respect to claims 12, 25, and 38.

Accordingly, it is respectfully submitted that the rejection of claims 12-13, 25-26, and 38-39 as anticipated by the Andrei patent should be reversed.

B. Conclusion

Each of the rejections set forth in the Final Office Action is improper and should be reversed.

Respectfully submitted,

/Janaki K. Davda/

Janaki K. Davda

Dated: July 9, 2007

Reg. No. 40,684

Konrad Raynes & Victor LLP

315 South Beverly Drive, Ste. 210

Beverly Hills, California 90212

Tel: 310-553-7977

Fax: 310-556-7984

## VIII. Claims Appendix

1. (Previously Presented) A method for processing predicates in an iterator function, comprising:
  - under control of an iterator function processor that executes as part of a data store engine, when an iterator function included in a statement is invoked,
    - obtaining one or more predicates included in the statement;
    - applying the one or more predicates to a row of data;
    - if applying the one or more predicates results in a match, returning the row of data; and
    - if applying the one or more predicates does not result in a match, searching for another row of data for which application of the one or more predicates results in a match.
2. (Previously Presented) The method of claim 1, wherein obtaining the one or more predicates comprises:
  - obtaining a qualification descriptor that describes the one or more predicates and one or more functions that are used to process the one or more predicates.
3. (Original) The method of claim 2, wherein each function is used to process one of the predicates.
4. (Original) The method of claim 1, wherein each of the one or more predicates comprises a simple predicate.
5. (Previously Presented) The method of claim 1, wherein the iterator function is invoked by the data store engine and further comprising:
  - returning the row of data to the data store engine.
6. (Previously Presented) A method for processing predicates, comprising:
  - under control of a data store engine,

receiving a statement including an iterator function and one or more predicates;

creating a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates; and

invoking the iterator function one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine.

7. (Original) The method of claim 6, wherein the qualification descriptor provides a handle to each of the one or more functions.

8. (Previously Presented) The method of claim 6, further comprising:  
when the iterator function is invoked, receiving an indication from the iterator function indicating which of the one or more predicates were applied by the iterator function.

9. (Original) The method of claim 6, further comprising:  
when the iterator function is invoked, receiving a row of data from the iterator function that matches the qualification of the one or more predicates.

10. (Original) The method of claim 9, further comprising:  
applying one or more additional predicates to the received row of data, wherein the one or more additional predicates refer to a column of data that is not in a result set generated by the iterator function.

11. (Original) The method of claim 9, further comprising:  
applying one or more additional predicates to the received row of data, wherein the one or more additional predicates performs a join between two tables.

12. (Previously Presented) A method for processing predicates, comprising:  
under control of a data store engine,  
receiving a statement including an iterator function and one or more  
predicates;  
creating a qualification descriptor that describes the one or more  
predicates and one or more functions that are to be used to evaluate the one or more  
predicates; and  
invoking the iterator function;  
under control of an iterator function processor,  
retrieving the qualification descriptor;  
obtaining a row of data that matches at least one of the one or more  
predicates in the qualification descriptor;  
setting indicators to indicate which of the one or more predicates have  
been applied by the iterator function processor; and  
returning the row of data to the data store engine; and  
under control of the data store engine,  
applying any of the one or more of the predicates included in the statement  
that have not been applied by the iterator function processor.

13. (Original) The method of claim 12, wherein the qualification descriptor  
describes one or more simple predicates to be applied by the iterator function processor.

14. (Previously Presented) An article of manufacture including a program for  
processing predicates in an iterator function, wherein the program causes operations to be  
performed, the operations comprising:

under control of an iterator function processor that executes as part of a data store  
engine, when an iterator function included in a statement is invoked,  
obtaining one or more predicates included in the statement;  
applying the one or more predicates to a row of data;

if applying the one or more predicates results in a match, returning the row of data; and

if applying the one or more predicates does not result in a match, searching for another row of data for which application of the one or more predicates results in a match.

15. (Previously Presented) The article of manufacture of claim 14, wherein operations for obtaining the one or more predicates further comprise:

obtaining a qualification descriptor that describes the one or more predicates and one or more functions that are used to process the one or more predicates.

16. (Original) The article of manufacture of claim 15, wherein each function is used to process one of the predicates.

17. (Original) The article of manufacture of claim 14, wherein each of the one or more predicates comprises a simple predicate.

18. (Previously Presented) The article of manufacture of claim 14, wherein the iterator function is invoked by the data store engine and wherein the operations further comprise:

returning the row of data to the data store engine.

19. (Previously Presented) An article of manufacture including a program for processing predicates, wherein the program causes operations to be performed, the operations comprising:

under control of a data store engine,

receiving a statement including an iterator function and one or more predicates;

creating a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates; and

invoking the iterator function one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine.

20. (Original) The article of manufacture of claim 19, wherein the qualification descriptor provides a handle to each of the one or more functions.

21. (Previously Presented) The article of manufacture of claim 19, wherein the operations further comprise:

when the iterator function is invoked, receiving an indication from the iterator function indicating which of the one or more predicates were applied by the iterator function.

22. (Original) The article of manufacture of claim 19, wherein the operations further comprise:

when the iterator function is invoked, receiving a row of data from the iterator function that matches the qualification of the one or more predicates.

23. (Original) The article of manufacture of claim 22, wherein the operations further comprise:

applying one or more additional predicates to the received row of data, wherein the one or more additional predicates refer to a column of data that is not in a result set generated by the iterator function.

24. (Original) The article of manufacture of claim 22, wherein the operations further comprise:

applying one or more additional predicates to the received row of data, wherein the one or more additional predicates performs a join between two tables.



25. (Previously Presented) An article of manufacture including a program for processing predicates, wherein the program causes operations to be performed, the operations comprising:

- under control of a data store engine,
  - receiving a statement including an iterator function and one or more predicates;
  - creating a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates; and
  - invoking the iterator function; and
- under control of an iterator function processor,
  - retrieving the qualification descriptor;
  - obtaining a row of data that matches at least one of the one or more predicates in the qualification descriptor; and
  - setting indicators to indicate which of the one or more predicates have been applied by the iterator function processor; and
- returning the row of data to the data store engine; and
- under control of the data store engine,
  - applying any of the one or more of the predicates included in the statement that have not been applied by the iterator function processor.

26. (Original) The article of manufacture of claim 25, wherein the qualification descriptor describes one or more simple predicates to be applied by the iterator function processor.

27. (Previously Presented) A computer system having at least one program for processing predicates in an iterator function, comprising:

- under control of an iterator function processor that executes as part of a data store engine, when an iterator function included in a statement is invoked,
  - obtaining one or more predicates included in the statement;
  - applying the one or more predicates to a row of data;

if applying the one or more predicates results in a match, returning the row of data; and

if applying the one or more predicates does not result in a match, searching for another row of data for which application of the one or more predicates results in a match.

28. (Previously Presented) The computer system of claim 27, wherein obtaining the one or more predicates comprises:

obtaining a qualification descriptor that describes the one or more predicates and one or more functions that are used to process the one or more predicates.

29. (Original) The computer system of claim 28, wherein each function is used to process one of the predicates.

30. (Original) The computer system of claim 27, wherein each of the one or more predicates comprises a simple predicate.

31. (Previously Presented) The computer system of claim 27, wherein the iterator function is invoked by the data store engine and further comprising:

returning the row of data to the data store engine.

32. (Previously Presented) A computer system having at least one program for processing predicates, comprising:

under control of a data store engine,

receiving a statement including an iterator function and one or more predicates;

creating a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates; and

invoking the iterator function one or more times, until receiving a done indicator from the iterator function, wherein each invocation of the iterator function

results in receiving either a row of data for which at least one predicate has been applied or the done indicator and wherein one or more of the predicates included in the statement that have not been applied by the iterator function are applied by the data store engine.

33. (Original) The computer system of claim 32, wherein the qualification descriptor provides a handle to each of the one or more functions.

34. (Previously Presented) The computer system of claim 32, further comprising:

when the iterator function is invoked, receiving an indication from the iterator function indicating which of the one or more predicates were applied by the iterator function.

35. (Original) The computer system of claim 32, further comprising:

when the iterator function is invoked, receiving a row of data from the iterator function that matches the qualification of the one or more predicates.

36. (Original) The computer system of claim 35, further comprising:

applying one or more additional predicates to the received row of data, wherein the one or more additional predicates refer to a column of data that is not in a result set generated by the iterator function.

37. (Original) The computer system of claim 35, further comprising:

applying one or more additional predicates to the received row of data, wherein the one or more additional predicates performs a join between two tables.

38. (Previously Presented) A computer system for processing predicates, comprising:

under control of a data store engine,

means for receiving a statement including an iterator function and one or more predicates;

means for creating a qualification descriptor that describes the one or more predicates and one or more functions that are to be used to evaluate the one or more predicates; and

means for invoking the iterator function; and

under control of an iterator function processor,

means for retrieving the qualification descriptor;

means for obtaining a row of data that matches at least one of the one or more predicates in the qualification descriptor;

means for setting indicators to indicate which of the one or more predicates have been applied by the iterator function processor; and

means for returning the row of data to the data store engine; and

under control of the data store engine,

means for applying any of the one or more of the predicates included in the statement that have not been applied by the iterator function processor.

39. (Original) The computer system of claim 38, wherein the qualification descriptor describes one or more simple predicates to be applied by the iterator function processor.

IX. Evidence Appendix

None

X. Related Proceedings Appendix

None